DECUS
PROGRAM LIBRARY

DECUS NO.	8-639
TITLE	OS/8 DISASM
AUTHOR	John E. Curtis
COMPANY	Curtis Institute East Moriches, New York
DATE	May 21, 1973
SOURCE LANGUAGE	PAL-8

50% for the listing and 100% in the source form outputs. Form feeds are also converted to multiple line feeds if T is specified. (This can be suppressed by a patch mentioned in Appendix D.)

The D option specifies source type output. Line numbers, addresses, instruction codes, address comments, and literal dumps are deleted.

The E option enables the EAE instruction disassembly.

DISASM output is paged at each new origin. Under certain circumstances pages with very little information are output and paper is wasted. However, pagination at each origin permits removal of overlay and literal pages for comparison with the affected code and so aids interpretation of the code.

3. Output Forms:

It is assumed that the reader is familiar with PAL8 source and listing formats. DISASM imitates these. In most cases the DISASM source output is not identifiable from an original PAL8 source except for the occurence of certain special neomonics used to make the output more easily understood. These special symbols are listed in Appendix A. The listing form adds line numbers and address comments to the usual PAL8 listing format. For easy editing, the line numbers are reset at the start of each page. The line numbers of the listing correspond to the positions in the source and may be used in editing the source. Because of the deletion of literal dumps, the page numbers do not correspond.

In the absence of defined symbols, the listing form might appear:

```
 27 12451  4567          JMS I   2567
 28 12452  1257          TAD     .+5
 29 12453  5454          JMP I   .+1
 30 12454  1356          TAD     2556
```

By the definition of the appropriate symbols, the same code might be listed:

```
 27 12451  4567          JMS I   (OUTPUT
 28 12352  1257 RETURN,  TAD     SAVAC
 29 12453  5454          JMP I   .+1
 30 12454  1356          TAD     2556    /EXIT
```

or better:

```
 27 12451  4567          JMS     OUTPUT
 28 12452  1257 RETURN,  TAD     SAVAC
 29 12453  5454          JMP I   .+1
 30 12454  1356          EXIT
```

OS-8 DISASM

1 Introduction:

DISASM is a program to convert an absolute binary file and a user symbol table into listing type or source type output. In producing a listing, emphasis is placed on making the listing as understandable as possible. In producing a source type output, emphasis is placed on duplicating as closely as possible the way an advanced programmer would write the source.

DISASM will recreate direct off-page references, local and zero page literals, symbolic address and data tables, and suppressed origins. These features are invoked by suitable symbol definitions. Displaced addresses such as .+5 and SYMBOL-6 are a standard part of DISASM.

DISASM can easily be used to create a source file when the binary tape and a listing are available. It is also designed for the disassembly of undocumented binary programs. The symbol table can be fully defined by inspection of a listing or gradually evolved from inspection of successive disassembly listings.

Included with DISASM is a program, SPLIT, for splitting large binary files into small segments for individual disassembly.

2. Loading and Calling DISASM:

DISASM is loaded and called in the usual manner with no special qualifications. A typical sequence of loading instructions would appear:

```
.R ABSLDR
*PTR:$
.SAVE SYS:DISASM
.R DISASM
*OUTPUT.PA<SYMTBL.PA,BIN01,BIN02,BIN01/T/D/E
```

The default output is LPT: but the T option must be used with most printers as, without it, the output contains tabulation codes. (The author has no line printer but uses the handler FORMAT under the device name LPT: Tabs should be set at the normal eight column intervals.)

The first input must be a symbol table or null. The Symbol table extension must be specified. The default extension is BN. Up to eight binary input files may be specified.

The T option causes tabulation codes in the output to be converted to spaces. This increases file length about

All symbol definitions for DISASM are made according to the field in which they appear. Labels appearing in the label column and in the address (operand) column are only those defined for the current memory field. Addresses are output according to the following priority:

1. exact symbol labels.
2. current address with a displacement up to 7 (.-7 to .+7).
3. symbols with displacement up to 7.
4. absolute octal addresses.

In the listing form only, if a label has value equal to the instruction, it is added as a comment. If no label in the current field has the correct value, but a symbol defined in some other field does, it will be printed.

Disassembly of instructions can be suppressed for one or more locations by the use of special symbols. The instruction can then be interpreted as an address or data. If address is specified, the symbol with smallest displacement (up to 7) in any field is entered in the operator column. Priority goes to the current field. If no symbol qualifies, an octal value is output. In the last example, the special symbol /@S=2454 caused "EDIT" to be printed; EDIT was defined as 1356.

4. DISASM Symbol Tables:

DISASM contains all the permanently defined symbols of PAL8 with the exception of IOT and OPR. It also contains a set of EAE symbols and the basic floating point symbols. The former are enabled by the E option and the latter by the definition of FLPNT, the zero page location of the floating point routine entry address. FLPNT must be the first symbol defined. The location and structure of the permanent tables are specified in Appendix C.

All of field 1 except the top page is devoted to user symbols. Six words are used for each symbol, so DISASM has capacity for 661 symbols. Programs using more symbols are disassembled in segments using only part of the total symbol table at any given time. The author usually defines symbols in several files. Separate files are used to define those symbols which are clearly local to a segment of code. The files required for several segments are merged using PIP before those sections of code are disassembled.

The user must define symbols for all labels, literals, and data or symbol items and tables. Special symbols are also available for introducing suppressed origins and altering instructions. All these user symbols must be input in the first input file specification, in one file.

The symbol file must start with a title. The first character in the file must be a "/". Up to 33 characters are taken from the first line for the title. The rest of the line is discarded.

If the floating point instructions are to be recognized, the second line must contain the definition FLPNT=n where n is the floating point indirect entry address. This definition serves only to define FINT=JMS I n. FLPNT should be redefined in each field in which it is used.

Labels must be defined by the field in which they occur. All labels for the same field must be defined contiguously. The field declaration has the form /FIELD=n where n is an octal digit. The declaration sets pointers to the first symbol in each field. Repetition of the same field declaration is a fatal error.

The labels or user symbols are usually strings of printing characters (including spaces) followed by an "=" and an octal value or address of not more than four digits. Note that the equal sign cannot be included in a symbol. The symbol "@" will be stored, but will not print back. Normally, each symbol is defined on a different line. See Appendix B for the technical description of symbol definition. DISASM will accept up to ten characters for a symbol. It ignores any additional characters. It will ignore, not store, any symbols with zero value. The comment slash is a legal character in symbols, but if it immediately terminates a value, it is recognized as a comment initiator and the rest of the line is ignored. If it is desired to temporarily delete a symbol, this can be done by inserting "0/" following the "=". With the exception of unusual user symbols and literal symbols, the file is acceptable to PAL8. The symbols not acceptable to PAL8 can be placed at the end of the file, separated from the others by "$=0". PAL8 can then be used with the N option to alphabetically list the symbols.

5. Special Symbols:

It has been noted that FLPNT has special meaning when it is the first symbol and that /FIELD has special meaning. DISASM recognizes a number of other special symbols.

All symbols starting with the character [are considered to be zero page literals. Under the D option, source type output, locations with labels starting with [and origins preceeding them are deleted.

All symbols starting with < are recognized as local, non-zero page, literals which may have been created by direct off-page references. The character < never appears in the output. If these symbols are referenced directly, < is changed

to (, but if referenced indirectly, the I and the < are deleted. Symbols starting with (are also recognized as local literals. These should be used where the value of the literal corresponds to a local or zero page address. As in the case of zero page literals, they are deleted from source type listings, but origins preceeding local literals are replaced by the pseudo-operator PAGE.

The symbol definitions /@SS=n and /@S=n suppress instruction disassembly and cause DISASM to output symbols instead. The definition /@S=n causes the instruction at address n to be interpreted as an address or symbol and then restores disassembly at the next location. The symbol /@SS switches DISASM into symbol table mode. /@S can be used to switch DISASM for one location only or to terminate a table.

The definitions /@NS=n and /@N=n switch DISASM to data mode in which the octal values are output in the operation column. /@NS is used to start a data table. /@N is used to insert a single value or to terminate a table. The /@S symbols take precedence and can be used to insert labels into data tables.

At times it is desired to alter the value of an instruction. I/O errors can be corrected and code altered. The definition /@I=m=n will cause disassembly of the code m at location n. In the listing form output, the instruction code in the third column is not changed, but the output in subsequent columns corresponds to the code m.

The definition /@O=m=n causes output of a suppressed origin of value m at address n. The current address is also reset. The new origin causes output of a new page. See the section on disassembling overlays.

The special /@ symbols are executed as part of the symbol table look-up routine and the search continues following their identification. Location of other symbols with the desired value terminates the search. Thus if a label is defined with the same value as a special symbol, the special symbol must preceed the label. Also if two labels are defined with the same value in the same field, the second will never be identified. It merely consumes table space and adds to the search time.

6. Using DISASM with a Listing:

DISASM was originally written to create source files for programs where source tapes acceptable to the system were not available. (FOCAL source was available only on DECtape.)

The symbols are copied directly from the listing. The symbol table is of some value, but it does not indicate the field in which the label belongs. The meaning of the

literals is identified from the text and the dumps are used as a check.

It is possible to copy a symbol file from the listing and immediately output a final source file. The probability of typing errors and the relative ease with which they can be detected from a listing type output make it desireable to first output the disassembly as a listing. The listing line numbers can also be used in editing the source output.

7. Using DISASM with a Symbol Table:

When a symbol table is available, but no listing, and it is desired to create a listing or source, it is usually easiest to first obtain a disassembly with no symbol table or better with one that contains only a title and a dummy field setting. This listing can be used to identify the fields in which the symbols are defined and to define the literal symbols. Some data and symbol tables can be identified from the first pass. The existence of zero page literals is indicated by a dump at the end of the code for a given field. The local or non-zero page literals are identified by an origin near the top of the page, code in every location to the top of the page, and normally another origin following the literal dump. The identification of the dump can only be confirmed by checking the sequence of references to the dump, but this is rarely needed. In the first disassembly, the dump might appear:

```
  3                *575
  4 00575  1426           TAD I   26
  5 00576  0453           AND I   53
  6 00577  0215           AND     415
```

The next page would start with an origin. The following definitions would then be added to the symbol table:

```
/@SS=575
/@S=577
<1426=575
(453=576
<215=577
```

Suppressing disassembly and outputting symbols conceals the rare ocassions when literals consist of instructions such as (JMS OUTPUT, but contribute more to identifying their meaning. On the second listing, the dump might appear:

```
  3                        *575
  4 00575  1426 <1426,     ERROR+2
  5 00576  0453 (453,      OUTPUT
  6 00577  0215 <215,      215
```

<1426 can now be redefined as <ERROR+2 and (453 as (OUTPUT. Note that the character < was used except in the case of (453.

453 is a current page address. Literals corresponding to current page or zero page addresses should be defined using (. If the data field is not current, TAD OUTPUT, obtained using <OUTPUT, is not executed in the same manner as TAD I (OUTPUT obtained by defining (OUTPUT.

7. Disassembling Undocumented Tapes:

In disassembling all but the shortest undocumented tapes, it is very valuable to use SPLIT. Frequent disassembly of short segments of the program with the symbol table freshly revised greatly assists interpretation of the code.

Following the first disassembly, using a symbol table which contains only a title and a dummy field setting, all literal symbols should be defined as above. DISASM cannot tell the user what the code does, so there is no alternative to tracing out the operation of the instructions. A quick scan of the first dump may reveal familiar I/O routines, USR calls, and similar routines which can quickly be labeled. In inventing new labels, it is important to avoid duplication. It is usually simplest to build the symbol table in segments. Segregate literals at the end. Editing and correcting the table will be easiest if the symbols are entered in order of increasing value within each segment. Use PAL8 with the N option to frequently list the symbols in alphabetical order.

8. Overlays and Patches:

Overlays and patches can be extremely difficult to interpret in their original location. If it is found that a section of code from 543 to 602 is to be moved to 200 to 237, it is very difficult to recognize the significance of an instruction such as JMP I 430 as meaning jump to the address specified in 573. Interpretation can be greatly simplified by inserting a suppressed origin using the definition /@O=200=543. The address will then appear at 230 and the instruction will be disassembled as JMP I 230. Labels for the overlay can then be defined using the translated values, in the interval 200-237.

There are no special precautions required in inserting suppressed origins in DISASM listings. References to labels outside the ovelay, even to literals, will be correct. However, PAL8 imposes severe restrictions on the use of literals in code displaced by suppressed origins and in code on the same page but preceeding the suppressed origin. After interpretation of the overlay, it may be necesary to delete the suppressed origin and define logical expressions to translate the address references. These can later be edited into the source output.

9. Floating Point Disassemblies:

As noted previously, if the first symbol is FLPNT=n, then JMS I n will be regarded as a command to enter floating point disassembly mode. If n=7, as is common, then on every occurance of 5407, DISASM will enter this mode. If the entry is data, then all code until the next zero will be incorrectly disassembled. Once this has been spotted, it can be suppressed by using a /@S or /@N symbol to suppress the disassembly of the 5407 data item.

Certain programs, such as FOCAL use non-standard floating point operation codes. With these programs, redefinition of the codes requires modification of the floating point table at 2054. If FEXT is not zero, a major patch will be required since exit requires calling a special routine. See Appendix c for the structure of the tables.

Special commands with the zero operation code such as square root have not been defined. They are not uniform in usage.

10. DISASM Error Routines and Codes:

If the output file becomes full, DISASM prints "FULL" and calls the command decoder. The user should input only an output file. Input files will be ignored. The first output file ends in an incomplete line. If merged using EDIT, no information is lost. If the first file is moved in anything except image mode, there is risk that the partial line will be lost.

All other detected errors are fatal and output files are lost. The codes for the errors are:

Output handler error: 0
Input handler error: 1
Output open error: 2
Output close error: 3
Default LPT: not available: 4
Symbol table error: 5

ISASM attempts to recover from most symbol table errors. The only errors causing error exits are:

Failure to start with a / (no title).
Initial field not specified.
Field specification repeated for the same field.
Symbol table overflow.

11. SPLIT:

SPLIT is a simple, not very elegant, file splitting program. Loaded and called in the usual manner, it expects a four character output name and a single input file. If only an output device is specified, it will supply the name SPLT.

SPLIT scans the input file and outputs it to a sequence of files sequentially numbered, eg. SPLT01.BN, SPLT02.BN, ..., SPLT99.BN. Whatever the output name the user specifies, SPLIT supplies the fifth and sixth characters and the BN extension.

Each time that SPLIT encounters an origin that is on a different memory page than the last, it closes the last file and inserts the new origin in a new file. If SPLIT has not seen any field setting, it does not insert any field setting. On the other hand, if it has seen a field setting, it starts every file with a field setting.

Since SPLIT will have closed a number of output files before it can detect a check sum error, it finishes all output and then reports the error. Other errors are immediately fatal. They include the same codes 0-3 as DISASM but 4 represents an improper end of file.

APPENDIX A
Symbols not in PAL8

In addition to the symbols defined in PAL8, DISASM uses the following symbols which may have to be defined before the DISASM output can be reassembled.

Micro Instructions, always available:

STM2=7344	STP3=7325	ST2K=7332
STM3=7346	STP4=7307	ST4K=7330
STP2=7305	STP6=7327	ST6K=7333

EAE Instruction, available with the E option:

MUY=7405	ASR=7415	SCL=7403
DVI=7507	LSR=7417	SCA=7441
NMI=7411	MQL=7421	MQA=7501
SHL=7413		

Floating Point Instructions, when FLPNT is the first symbol:

Regular: FINT=JMS I FLPNT FEXT=0

MRI Instructions:

FADD=1000	FMPY=3000	FGET=5000
FSUB=2000	FDIV=4000	FPUT=6000
FNOR=7000		

APPENDIX B
User Symbol Input

Considerable power has been introduced into DISASM as it evolved by manipulation of symbols. Many such opportunities have already been formalized, but the user who understands the symbol input algorithms can still invent new tricks. The format of the input is much more flexible than indicated. The PAL8 symbol table can be used as input if a field setting is inserted and all spaces in the table are converted to tabs. The input routine first tests for a /. If this is found, the first line is accepted as a title (else fatal error). The input routine then requests the first definition and tests for FLPNT. If present, it is processed, and another definition requested. In either case, the routine then enters a loop to accept a definition, test for a /FIELD symbol, if recognized, set the pointers and go back for another definition, else test the value for zero. If zero, go back for another, else store it.

The definition fetching routine calls a symbol fetching routine. If the symbol is a /@ special symbol, a special routine is called to assemble the values for the special symbol. (Special symbols are flagged by the first word being zero. As they are identified, a check is also made for symbols starting @@ and they are converted to @A to avoid conflict.) The definition fetching routine then calls a subroutine to fetch the value. The special symbols are stored as a zero, then a negative symbol number, and then a special value if needed.

The symbol fetching routine recogizes 275 (=) and all codes below 240 as terminators. When first called, the routine skips terminators. The first non-terminator starts the character scan. The first ten characters are packed as 6-bit chopped ASC II Additional characters are scanned and discarded until a terminator is found. If a terminator is found before the tenth character, the storage is filled out with zeroes (@). In printing back symbols, DISASM ignores zeroes and so does not print back the character @. The normally available terminators are horizontal tab, line feed, vertical tab, form feed, carriage return, and equals.

The value accepting routine skips symbol terminators. The first non-sysmbol-terminator initiates value acceptance. Only octal digits are accepted. Up to four of these are converted to a binary value. Any other character terminates the value. In case there are four octal digits, the next character is fetched for use as a terminator. The terminator is tested for /, and if found, the text to the next carriage return is dumped.

The definition "SYM=1" followed by a carriage return will pack 2431,1600,0,0,0 for the symbol and 0001 for the value.

The definition "SYM=AB" will cause the input value to be started and terminated by the "A". The value will be zero and SYM will not be stored. The input routine will then try to fetch a value for the symbol "B".. Thus:

```
SYM=AB
NEXT=5
```

will store only "EXT" with a value of 5. The "A" and the "N" are lost as value terminators. This is the result of the routine to recognize the comment slash.

APPENDIX C
DISASM Permanent Tables

Two different types of tables are used in DISASM. The first contains instruction codes each followed by the two word (4 chopped character) neomonic. Padding is done with spaces in the permanent tables. The first contains entries in an order such that combined micro-instructions precede their components. They are searched under a mask and matching bits are deleted under the mask. These tables are terminated by zero entries. The second type is accessed by displacement into the table and contains only the neomonic.

I/O Tables: These are accessed indirectly using a displacement address table. The device number (bits 3-8) is used as a displacement into a table at 2400. Zero entries indicate that the device is not defined. Non-zero entries are addresses of tables of the first type containing the micro instructions for the device. The device tables extend from 3263 to 3466. 3467 to 3577 is available for expansion. If more space is required, the user might consider disabling the EAE or floating point commands to obtain more space. Provided that the table at 2400 is properly maintained, rearrangement of the I/O table area is simple.

Micro Instructions: The tables for Group 2 (3033), Group 1 (3160) and the EAE (3116) instructions are of the same format as the I/O tables. The EAE instruction table is easily altered. The Group 2 table has a special first entry wich must be preserved.

Operands: The regular operands (AND, TAD, ISZ, DCA, JMS, and JMP) are in a displacement table at 3102. The floating point operands are in a displacement table at 2054. Since this table does not contain a zero entry, its base is 2052.

APPENDIX D
Possible Patches

A few hints are provided on patching DISASM for special applications.

Output control: Output is designed for the OS-8 handler FORMAT. This handler is used under the device name LPT: on the system under which DISASM was developed. It may desired to complement the T option or to change the relation of the expansion of tabs and form feeds. As loaded, DISASM is set for both tab and form feed conversion. The T option patches the code to suppress conversion. The option can be complemented by changing location 3651 from SZA CLA to SNA CLA (7650). If form feed terminals are standard, changing 3652 to 5255 will cause the program to always output 214 codes, T option or not. Changing 3654 to 7200 would cause tabs to be always converted.

Special Micro Instructions: The STM2 and similar combined micro-instructions are included to improve interpretation. They can be deleted by replacing thier operation codes by 4000. (Zero would terminate the table.) The new DEC symbols cannot be implemented as they are too long for the operation output routine.

Alteration of the floating point MRI instructions requires only changing the neomonics in the table. Changing FEXT would require reassembly (get the listing and use DISASM to disassemble DISASM).

Suppressing literal recognition does not require a patch. Prefix the literal with 0.

Patch to correct internal buffer overflow problem:

LOC	00127	Change to 7601 (was vacant)
LOC	00226	Change to 1127 (was 1173)